REMARKS

In the Office Action mailed March 4, 2005, the Examiner noted that claims 1-11 were pending, claim 12 had been withdrawn and rejected claims 1-11. No claims have been amended, and, thus, in view of the forgoing claims 1-11 remain pending for reconsideration which is requested. No new matter has been added. The Examiner's rejections are traversed below.

In the Action, on pages 2 and 3, the Examiner rejected claims 1-11 under 35 USC section 112 as not being supported by the application as filed. The Examiner pointed to particular phrases in the claims. It is submitted that these phrases of concern are supported by application pages 19-24 and figures 2-5. Withdrawal of the rejection is requested.

On page 3 of the Action the Examiner rejected claims 1, 2 and 5-11 as obvious over Serbinis and Usdin. On page 7 the Examiner rejected claims 3 and 4 as obvious over Serbinis, Usdin and Hashimoto.

The present invention involves a hierarchy of documents where a document in a higher level, say a master drawing, refers to a document in a lower level, say a component drawing. The Examiner compares this feature of the present invention with Serbinis in figure 3 and the associated text at col. 7, starting at line 16 and to Usdin on pages 126, 127 and 128.

Figure 3 of Serbinis has reference arrows indicating a reference from one level element to another level element that point only in one direction from the leaves of the tree, such as leaf 74A, to the root, 70, of the tree. That is, the reference arrows point from a lower level to a higher level. The text of Serbinis noted by the Examiner particularly states:

> Referring now to FIG. 3, an illustrative hierarchical storage scheme for storing electronic documents on store 30 of DMS system 17 is described. Each user of DMS system 17 preferably has access to one or more document groups 70, where each document group comprises a collection of document objects 72A and 72B. One document may belong to one, many or no document groups. Each document group 70 has a name, a description, and a service defined type for defining the document type (e.g., word processing file). A document group may have one or more parent document groups. The document groups preferably have extensible property types.
>
> Document objects 72A and 72B represent a generalized high level description of a document, and consist of a document name. Document objects also may have extensible property types.
>
> Document instances 73A, 73B and 73C correspond to specific instances of a document, and each include details about the document, a reference to the document, a document state, description, size, priority, encryption type and expiry date. The default document states are "pending," "active," "archived" and

> "deleted." Document states are extensible by service. A document state log is kept to track when a document instance has changed state, as described hereinbelow.
> DMS system 17 also preferably supports multiple versions of documents, for example, versions 74A and 74B. A document version object is employed in document information tables 61 of database 25 and is used to maintain version relationships between document instances of a given document. Each document version instance 74A and 74B includes a reference to the parent and child document instance, a version name and a unique version ID.
> Document records are created in DMS database 25 the first time a new document is stored on DMS system 17. Document instance records are created when new documents or new versions of existing documents are stored to the DMS system. Document group records may be created when logical collections of documents are stored at the same time and it is desired to maintain the relationship between the documents. Also, according to the authorization information submitted by a document originator, new document rights, document group rights and document instance rights are created for the document. A document store record references a document instance and a store and includes a unique key/name to the document's storage location.

(See Serbinis, col. 7, lines 16-62)

This text of Serbinis indicates that the higher levels in the Serbinis data structure are not documents but essentially pointers to the documents (instances).

It is submitted that Serbinis does not teach or discuss documents in a higher level referencing documents in a lower level.

The text Usdin particularly states:

> XML assumes that data are hierarchically structured and that many types of useful and identifiable information contain other useful and identifiable types of information. The named types of information in XML are called elements. XML data are packaged into "document instances" (which most non-XMLers would call documents), which consist of a named "root" element which begins at the beginning of the document, contains the entire document, and ends at the end of the document. It is common for many of the elements that are inside the root element to contain other elements, which may in turn contain yet other elements. This hierarchical structure is fairly obvious in text documents: a section starts before the heading that starts that section and ends just before the next heading of the same level. A section may contain a heading, some paragraphs, and perhaps subsections that contain headings and paragraphs of their own. This hierarchical structure is very valuable in rearranging documents, or in cutting parts of one document out and inserting them into another; if a user wants to grab the part of a document that addresses a specific topic, s/he will want the heading plus all of the content until the next heading.
> Hierarchical structure may be less obvious in non-text documents, but even there it is usually useful to divide the information into a small number of groups, which are then subdivided. For example, a catalog order may consist of information about the customer, information about the items ordered, and information about payment. Each of these groups may contain additional details.

XML document instances are simple tree structures; each instance has one and only one root and is divided into non-intersecting branches. There are no loops in XML document structures.

(See Usdin, page 126)

XML provides a mechanism to specify what tags will be used in a class (type) of documents, what the permitted relationships are among those tags, what attributes are allowed on each tag, and what the values of those attributes may be. This information is encoded in a Document Type Definition (DTD). DTDs ma be used to drive SML-savvy tools that assist authors in creating documents: by leading them through the process of creating sequential information in the correct order; by providing menus of only those elements that are valid in a particular context; and by informing authors when documents are missing required information or otherwise violates the rules for a document type.

The syntax for DTDs specified in XML 1.0 is a subset of SGML's DTD syntax. A DTD can specify what the root element is for a document type, what elements may occur in the document type, what relationships are permitted among these elements, what attributes are allowed on each element, and what value these attributes may take. There is a limited capability for expressing element relationships and a limited capability to specify attribute content.

There are also several proposals for alternate and richer ways to define/describe the contents of XML documents. Among the features under discussion are the use of XML syntax for defining XML documents, richer constructs for modeling element relationships, the ability to define the contents of elements and to more precisely define the contents of attributes, and to define relationships among the contents of various elements. A work group to address these issues has recently begun to examine them, and it is likely that XML DTD syntax will be supplemented or replaced with a richer data definition language.

The XML Recommendation is the first member of the XML family to reach W3C recommendation status, but there are several proposed recommendations and W3C Notes related to XML, plus recommendations, notes, and standards from other bodies for applications of XML. These related specifications will provide shared ways to communicate how XML information should behave, be accessed, and be displayed. Their wide adoption will significantly improve the ease with which complex applications can accommodate unknown tags and tag structures. It is important to note that none of these related specifications addresses what tags users should use for the basic content of their information. However, some do provide semantics for specifying how links (such as hypertext pointers and targets) should be identified and for specification of style information.

XML—Extensible Markup Language (http://www.w7.org/WD/WD-xlink). The base document for the XML family of standards and for all XML applications.

Xlink—XML Linking Language (http://www.w3.org/TR/1998/WD-xptr-1998303). Additional functionality for advanced hypertext and hypermedia capability such as links to multiple destinations, bi=directional links, links annotating read-only documents, specification of link behavior, and databases of links. Xlink is based on, but not identical to, the hyperlinking functionality of HyTime, i.e., ISO/IEC10744.

The previous XLL work was split into Xlink and Xpointer (see below) to separate the linking functionality from the addressing of the objects being linked. See http://www.w3.org/TR/NOTE-xlink-principals for additional information on the design principles of both Xlink and Xpointer.

> Xpointer—XML Pointer Language (http://www.w3.org/WD/WD-xptr). Specification of constructs for addressing into the internal structures of XML documents. This includes absolute locations, such as the root of the documents or an element with a specified ID, and locations specified relative to known locations. The relative locators can navigate forwards, backwards, up and down through the element tree. Xpointers can specify spans as well as single points, and can address text regions as well as elements. Xpointers are based on, but are not identical to, TEI extended pointers.
>
> XSL—Extensible Style Language (http://www.w3.org/TR/WD-xsl). The language for stylesheets for XML documents. The processing model species a transformation stage to construct a "result tree" from the input "source tree"; then the styles are applied to the result tree. The result tree can be completely different from the source tree, and the result tree may use only parts of the source tree, re-use it all, or re-use it multiple times, as well as add generated structure that was not in the source tree.
>
> XML Namespaces (http://www.w3.org/TR/1998/WD-xml-names-19980801). A simple method for qualifying names used in XML documents by associating them with namespaces identified by a URI. Use of namespaces will promote re-use of markup, and will allow composite documents built from fragments from many sources, since the URI will disambiguate element types that would otherwise have the same name. Namespaces will also promote re-use of the software modules that process the markup, since a module can act on element type names within a known namespace and ignore all others.

(See Usdin, pages 127 and 128)

This text of Usdin describes the hierarchy of a document, that is, a hierarchy within a document and not a hierarchy of documents where a document at a higher level references a document at a lower level.

It is submitted that that Usdin also does not teach or discuss documents in a higher level referencing documents in a lower level.

As a result, it is submitted that the combination of Serbinis and Usdin does not teach or discuss documents in a higher level referencing documents in a lower level.

The Examiner is cites Hashimoto for its alleged teachings of updating a document. That is, Hashimoto (or the combination of Hashimoto with of Serbinis and Usdin) also does not teach or discuss documents in a higher level referencing documents in a lower level

That is, the prior art does not teach or suggest the feature of the present invention where a higher level document references a lower level document. Withdrawal of the rejection for this reason is respectfully requested.

The claims of the present invention also call for the documents in the upper and lower hierarchy layers to have version numbers. The Examiner points to Serbinis at col. 7, lines 51-57 for this feature. This text of Serbinis is set forth in the above quotation from Serbinis. This text calls for different versions but does not appear to discuss assigning a version number. Although the text at lines 47-49 mention a unique version ID for the versions 74, versions of documents can be tracked in many ways besides using a version number. A unique version ID need not be or include a version number. It could be a unique pointer. The location of a version in a tree, such as specified by a record pointer indicating a child or a parent record can signify a version relationship without using a version number. This text also says nothing about using a version number in a higher level of hierarchy. It is submitted that the prior art does not teach or suggest the feature of the version number in the lower and upper levels.

It is submitted that the claims satisfy the requirements of 35 U.S.C. section 112, paragraph 1. It is also submitted that the claims are not taught, disclosed or suggested by the prior art. The claims are therefore in a condition suitable for allowance. An early Notice of Allowance is requested.

If any further fees, other than and except for the issue fee, are necessary with respect to this paper, the U.S.P.T.O. is requested to obtain the same from deposit account number 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 6/7/5

By: ______________________
J. Randall Beckers
Registration No. 30,358

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501